Amendments to the drawings,

There are no amendments to the Drawings.

Remarks

Status of application

Claims 1-25 are pending in the current application. The Examiner's withdrawal of art rejections based on Sriram and Vaid, in response to Applicant's prior Amendment, is greatly appreciated. The claims stand rejected in view of new cited prior art. In view of clarifying amendments to the claims and remarks made below, re-examination and reconsideration are respectfully requested.

The invention

An e-mail system constructed in accordance with the present invention includes a composer module ("Composer"), a message transport agent (MTA), and a mass-mail accelerator (MMA). A concise summary of Applicant's claimed invention is provided in Applicant's prior Amendment.

Prior art rejections

Claims 1-25 stand rejected under 35 U.S.C. 103(a) as being unpatentable over extensive consideration of US Patent 5,937,162 to Funk. Regarding independent claims 1, 13, and 21, for example, the Examiner states the following basis for rejection:

> establishing a plurality of queues in the system, zero or more of these being specific queues for handling mail to a specific set of domains, and one being a general queue for transferring e-mail to domains not handled by specific queues, (Cal. 2, lines 1-22; Cal. 11, lines 36-67; Cols, 12-13; & Cal. 14, lines 1-45), (Examiner notes that Funk does not specifically enumerate a "general" queue for handling domains not handled by "specific" queues; how~ever the same would have been~ obvious to one of ordinary skill in the art at the time of invention by Applicant as Funk clearly teaches queuing by destination address to (special) designated hosts wherein any mail not sent to a specific designated host would obviously be placed in a "general" queue for continued processing. The motivation to "pre-sort" messages into "general" and "specific" queues is to speed up message delivery time, (Col. 11, lines 49-52), and reduce "clumping", (Col. 12, lines 30-34));
>
> receiving at the system a request to process for transfer a plurality of outbound e-mail message, each e-mail message specifying delivery to at least one recipient at a particular domain, (Col. 2, lines 1-22; Col. 11, lines 36-67; Cols. 12-13; & Col. 14, lines 1-45); and

for each given e-mail message, processing the given e-mail message by:

determining (pre-sort) what domain the given e-mail message is destined for, if the determined domain for the given e-mail message is a specific domain handled by a corresponding specific queue, assigning the given e-mail message to the corresponding specific queue for transferring the given e-mail to said specific domain, otherwise assigning the given e-mail message to said general queue, (Col. 2, lines 1-22; Col. 11, lines 36-67; Cols, 12-13; & Col, 14, lines 1-45); and

without waiting for confirmation that the given e-mail message has been successfully processed for transfer to another system, proceeding to process the next one of the e-mail messages, (Col. 11, lines 36-67; Cols. 12-14; & Col. 15, lines 1-3), (Examiner notes that the pre-sorting, queueup, hash and control processors clearly and obviously do not require confirmation of successful processing into particular queues).

However, the Examiner notes that Funk does not specifically enumerate a "general" queue for handling domains not handled by "specific" queues. Nevertheless, the Examiner contends that it would have been obvious to one of ordinary skill in the art at the time of invention by Applicant as Funk "teaches queuing by destination address to (special) designated hosts wherein any mail not sent to a specific designated host would obviously be placed in a "general" queue for continued processing." Although Applicant does not agree that Applicant's "general" queue feature is obvious in view of Funk, Applicant's claims have nevertheless been amended to distinguish Applicant's invention on a variety of additional grounds.

At the outset, it is important to note that Applicant's approach is a fundamentally different one, which will now be illustrated in further detail. In contrast to Applicant's invention, Funk's approach is to make a specific connection for a given large e-mail service (e.g., AOL, Hotmail, etc.). Accordingly, the Funk system is only optimized for a specific set of large e-mail services, and thus fails to optimize outgoing e-mail messages destined for lesser (smaller) domains. Importantly, as the majority of e-mail handled by a given e-mail server may in fact to be destined for these lesser domains, Funk's approach is significantly inferior to that provided by Applicant's system.

As noted by the Examiner, both Applicant's system and Funk's system attempt to

improve performance by assigning work to concurrently-operating (i.e., parallel) queues. In Funk, this parallelism is achieved by looking up the list of SMTP servers and their host names that receive e-mail for a specific domain (e.g., AOL). Consider, for example, Funk's actions when sending e-mail to the AOL e-mail service. Upon issuing a request for AOL e-mail servers, Funk's system will receive back a list (MX records) of e-mail servers (e.g., b.mx.aol.com, etc.), such as six particular e-mail servers servicing AOL. The Funk's system will establish a connection to each of those six e-mail servers, and "spread" (i.e., simple hash) e-mail destined for AOL evenly across those connections. In effect, the Funk system would open six connections to AOL and then begin essentially "jamming" all AOL-bound e-mail into those six connections.

Consider, in contrast, the operation of Applicant's system. In Applicant's system, a specific queue can be configured for AOL, similar to that described above for Funk. However, by virtue of Applicant's MMA, the load-balancing or parallelism achieved is not simply a static assignment of connections to e-mail servers for a specific domain. In Applicant's system (continuing with the above AOL example), upon arrival of a seventh e-mail message, if all of the existing six connections are still busy (i.e., busy handling delivery of prior e-mail messages), Applicant's system will create a <u>seventh</u> connection to one of the aforementioned six AOL e-mail (MX) servers, and will begin using that <u>dynamically-created</u> connection for delivering e-mail. This process (i.e., creating additional connections based on run-time dynamics) will continue in Applicant's system in order to reach an equilibrium state: any time a message arrives, there is at least one connection available. Whereas the Funk system would only create six connections (total) in this example, Applicant's system may create a multitude of connections (i.e., well beyond six) in order to keep the flow of e-mail through the system optimal. It should be noted that Applicant's system will not create more connections than necessary, unless it is configured to do so. On the other hand, Applicant's system will not limit itself to just one connection for each e-mail (MX) server of the e-mail servers reported for a given domain, such as AOL.

Like Funk' system, Applicant's system initially establishes connections to the six e-mail servers. However, from that point on, it should now be readily apparent that the

two systems diverge substantially. Importantly, Applicant's system will create multiple MTA (message transport agent) threads, such that <u>each</u> particular recipient e-mail server may in fact be connected or talking to <u>multiple</u> MTA threads at the same time (in stark contrast to Funk's one-to-one approach).

This distinction is brought to the forefront by Applicant's amended claims. Amended claim 1, for example, includes the claim limitation of (shown in amended form):

> establishing a plurality of queues in the system, zero or more of these being specific queues for handling mail to a specific set of domains, and one being a general queue for transferring e-mail to domains not handled by specific queues<u>, each said queue being configured to spawn a number of message transport agents (MTAs) for connecting to available e-mail servers for a given domain, wherein the number of MTAs spawned for a given domain is determined based on run-time dynamics, and wherein the number of MTAs spawned for a given domain may exceed the number of the available e-mail servers for that given domain;</u>

As recited, the claim limitation specifies that a number of message transport agents (MTAs) are spawned dynamically based on the load conditions encountered. And the number of MTAs spawned may in fact exceed the number of available e-mail servers for a given domain. This is not the same as Funk's approach of creating a static set of connections based on the number of available e-mail servers for a given domain (e.g., six connections, for six AOL MX servers).

Support for the foregoing amendment is found in Applicant's specification, at the paragraph beginning at page 21, line 14:

> **If, when processing an incoming message, a given queue thread finds all of its MTA threads busy, the queue thread may launch another MTA thread** (unless the queue thread has reached a user-specified maximum number of corresponding MTA threads). The newly created thread will then proceed to connect to the destination MTA and attempt delivery. **This process of spawning new MTA threads may continue until underlying resources of the base hardware system are exhausted** (e.g., system has run out of file descriptors or memory). In instances where no more MTA threads

> can be created, the system logs corresponding information to a log file, thereby allowing the system administrator to fine-tune the underlying system (e.g., adjust the balance of queue and MTA resources) for the next run.
>
> (Emphasis added)

As shown by the foregoing, if a queue (thread) discovers that its currently-available MTAs and associated connections are all busy (e.g., corresponding to the scenario of six busy AOL connections in the example above), Applicant's system will spawn another MTA that is capable of connecting (to one of the recipient e-mail servers) and is ready to accept work. Applicant's system increases parallelism, when compared to Funk: Funk is limited to 1:1 connections for recipient e-mail servers for the intended destination domain. Applicant's system, in contrast, spawns additional ones, as required to optimize system throughput.

This feature of Applicant's invention is further demonstrated in Applicant's specification at the paragraph beginning at page 29, line 2:

> **Suppose** in step 606 **that no MTA threads are available**. In that case, processing proceeds as shown in Fig. 7. Step 701 is shown to indicate that these method steps are invoked in the context of "no MTA threads ready." What happens at this point depends on the configuration of the queue, specifically, whether the system is allowed by limits imposed in the configuration to create any more MTA threads. If the system has reached the configuration-specified limit, the method simply blocks and awaits the availability of an MTA thread, as indicated by step 702. However, if it has not reached this maximum limit, **the method may proceed to step 703, to create or spawn a new MTA thread. In that case, after step 703, the method will assign the work (of message delivery) to the newly created MTA thread, as shown at step 704.** Additionally, the newly created thread becomes part of the general pool that this queue can use. As shown by the foregoing, **the system is able to tune itself based on run-time dynamics,** such that the system reaches equilibrium, or steady state, where it does not need to create any more threads and the ones that are there are usually busy.

The above paragraph describes Applicant's implementation of this feature in further detail: configuration possibilities when a specific queue when it discovers all of its

MTAs (MTA threads) are busy. The queue may be configured so that it is not allowed to create additional MTA threads (e.g., beyond some configured limit), in which case the queue will wait for an existing MTA thread to become ready (and then assign the work to that MTA thread). While waiting, Applicant's queue manager (thread) builds up a list of work that needs to be performed, subject to any configuration constraints imposed by the user (administrator). The above demonstrates how Applicant's system is able to tune itself by making a new connection in order to satisfy the workload that is being pushed on to the queue manager thread. This optimization, which is based on run-time dynamics, is not taught or suggested by Funk (and Funk's static assignment of connections teaches, if anything, away from Applicant's invention).

Furthermore, in stark contrast to Funk, Applicant's system <u>will not assign work</u> to an MTA thread (for delivery of an e-mail message to one of the MX servers for AOL) <u>until</u> that thread has announced that it is ready to accept more work -- which means (among other things) that a corresponding MTA at the remote end (i.e., AOL MTA) has also indicated that it is ready to receive more work.

The importance of the foregoing claim limitation is brought to light by the following claim limitation (shown in amended form) of claim 1:

> if the determined domain for the given e-mail message is a specific domain handled by a corresponding specific queue, assigning the given e-mail message to the corresponding specific queue for transferring the given e-mail to said specific domain, otherwise assigning the given e-mail message to said general queue, <u>each queue maintaining a "ready" list for assigning the given e-mail message to an MTA that has indicated that it is available for work,</u> [...]

As shown, work is assigned only to an MTA that has announced itself as ready for work. Such MTAs are maintained on a "ready" list.

Support for the claim amendments is found in Applicant's specification. For example, at the paragraph beginning at page 21, line 4, Applicant's specification states:

> During MMA operation, once a message has been passed to a queue, that queue examines its MTA threads to see if one is ready to accept the message.

> **If an MTA thread is ready, the queue will assign the message to that MTA thread** (which exists inside the MMA 500). **Once a message is assigned to an MTA thread, that thread is no longer available and, thus, it marks itself as "busy" (or otherwise removes itself from a "ready" list).** The MTA thread proceeds to handle the work of the SMTP exchange between the MMA and the target real-world MTA (e.g., AOL MTA). **While a given MTA thread is waiting for a reply from the destination MTA (e.g., AOL MTA), the MMA can proceed to do other work.** Thus, for instance, while a given message is being handled by a particular MTA thread, other incoming messages can be injected, queued, requeued, moved around, or the like, within the system.
>
> (Emphasis added)

As shown, the queue (which has a list of work to be done) maintains a list of MTAs (threads) that are "ready" -- that is, available to do work. Specifically, the queue doesn't simply maintain a list of of connections that it can push e-mail to (in direct contrast to Funk's approach). Instead, the queue is waiting for one or more MTA threads to announce availability for doing work. Any MTA that is ready is added to the "ready" list (as described in the specification passage above). In other words, the "ready" list represents the set of MTAs that are currently available to accept work -- rather than just things the e-mail system happens to have a connection to that may eventually be ready for work.

Funk lacks such sophistication. Funk instead opens a static number of connections, and then just proceeds to evenly spread outgoing e-mail among those connections. Funk's simplistic approach of evenly spread work among the recipient e-mail servers (i.e., in this example, 1/6 work for each of the six connections) fails to take into account the dynamics of the current e-mail environment. For example, it may be the case that one of the AOL e-mail servers responds more quickly than the others, and thus is in a position to take on a higher volume of work. Or, conversely, one of the AOL e-mail servers may fail, whereupon it he is unable to handle any volume of work. As should be readily apparent, Funk's approach does not take advantage of these dynamic factors to adjust load balancing in a manner that leads to optimal throughput of e-mail traffic. At best, the Funk approach leads to suboptimal load-balancing, under real-world

conditions.

With Applicant's approach, the e-mail system will give more work to any MTA thread that comes back faster and is ready sooner. As alluded to above, this also has ramifications in terms of error processing. For example, if one of the remote e-mail servers becomes unavailable or gets slow (and thus the corresponding MTA thread remains busy), Applicant's system will no longer channel work to that server/corresponding MTA (while it is slow or unresponsive). Instead, Applicant's system would only resume sending work to that server when it indicates that it is ready for more work (whereupon, the corresponding MTA thread becomes "ready").

The remaining independent claims (i.e., claims 13 and 21) have likewise been amended to explicitly recite the above-mentioned distinguishing features. For example, claim 13 includes the limitation of (shown in amended form):

> a plurality of queues for processing incoming e-mail messages, at least one queue being designated as a specific queue for processing e-mail messages destined for a specific domain, wherein the queues are dynamically configurable at runtime to increase throughput via spawning multiple connections to each e-mail server for said specific domain;

As shown, the foregoing amendment requires the recited queues to be dynamic in nature -- dynamically adjusting their run-time operation to increase throughput. Particularly, multiple connections may be spawned or created for each (recipient) e-mail server. Funk, in contrast, specifies a one-to-one (1:1) relationship between connections and e-mail servers (e.g., six connections for six e-mail servers, as discussed in the AOL example above). It is submitted that Funk does not teach the above claim limitation, and in fact his static approach teaches away from Applicant's dynamic approach.

Similarly, claim 21 has been amended to incorporate specific claim language setting forth these distinguishing features. For example, claim 21 includes the limitation of (shown in amended form):

> wherein each said queue is configured to assign an e-mail message to a

message transport agent (MTA) that is available for sending the e-mail message to a given domain, and is configured to create additional MTAs when none are available to accept work.

As shown, the recited queue elements are required to assign e-mail traffic to available MTAs, and in instances where none are actually available the queues will in fact proceed to create additional ones. As previously stated above, nothing in Funk can reasonably be construed to provide this dynamic operation. Yes, Funk does provide a form of load balancing. However, Funk's load balancing is statically configured up front; it does not automatically reconfigure itself based on run-time dynamics. Accordingly, it is respectfully submitted that Funk does not teach this claim limitation, and therefore the amended claim distinguishes over Funk.

For the reasons stated above, it is respectfully submitted that Applicant's amended independent claims all distinguish over Funk. The dependent claims, although not explicitly discussed, are believed to be allowable by virtue of dependency from Applicant's independent claims 1, 13, and 21, as discussed in detail above. Accordingly, Applicant believes the above-mentioned amendments clearly distinguish Applicant's claimed invention from all art of record, and that the rejection under Section 103 is overcome.

Conclusion

In view of the foregoing remarks and the amendment to the claims, it is believed that all claims are now in condition for allowance. Hence, it is respectfully requested that the application be passed to issue at an early date.

If for any reason the Examiner feels that a telephone conference would in any way expedite prosecution of the subject application, the Examiner is invited to telephone the undersigned at 408 884 1507.

Respectfully submitted,

Date: November 16, 2005

Signature Valid

Digitally signed by John A. Smart Date: 2005.11.16 16:52:30 -08'00'

John A. Smart; Reg. No. 34,929
Attorney of Record

408 884 1507
815 572 8299 FAX